Econometric Institute Report No. 9707/A

NEW VARIANTS OF FINITE CRISS-CROSS PIVOT ALGORITHMS FOR LINEAR PROGRAMMING

Shuzhong Zhang*

February, 1997

ABSTRACT

In this paper we generalize the so-called first-in-last-out pivot rule and the most-often-selected-variable pivot rule for the simplex method, as proposed in Zhang [13], to the criss-cross pivot setting where neither the primal nor the dual feasibility is preserved. The finiteness of the new criss-cross pivot variants is proven.

Key words: Linear programming, criss-cross pivot method, anti-cycling.

AMS subject classification: 90C05.

---

* Econometric Institute, Erasmus University Rotterdam, The Netherlands.

1 Introduction

In this paper we introduce two new variants of the finite criss-cross pivot method for linear programming. In the first new variant we use exclusively the so-called *first in last out* (and symmetrically, *last out first in*, refering respectively to the leaving/entering basis variable selections) rule in selecting the pivoting elements. The second new variant uses a similar, but different, pivot rule, viz. the *rule of selecting the element that has been selected most often*. These two pivot rules were introduced in their explicit form as anti-cycling simplex pivot rules by Zhang [13]. The rules are simple and easy to remember. We shall see in this paper that they generate new finite criss-cross pivot algorithms as well.

The criss-cross method was first introduced by Zionts [14] in 1969 as a pivot algorithm for solving linear programming requiring no feasibility of the basis. About ten years later, inspired by Bland's smallest subscript pivot rule for the simplex method, a finite criss-cross pivot algorithm was independently proposed (with different motivations) by Chang [2], Terlaky [9] and Wang [12]. That algorithm works in a remarkably similar fashion as the smallest subscript pivot rule of Bland [1] for the simplex method. The difference is that no feasibility is required to be maintained in the pivoting procedure. In this sense, the criss-cross method is not a simplex type method. One big advantage of the criss-cross method is that no phase-I procedure is needed: Any basis will be equally fine to start with. It was shown that the algorithm finds an optimal basis, or concludes that either the primal or the dual problem is infeasible, in finite amount of pivot iterations; see [9] and [12]. Moreover, the method was generalized to solve the so-called oriented matroid programming problems ([10] and [12]). A thorough survey on the historical account and the recent developments about the criss-cross method can be found in Fukuda and Terlaky [7]. For a survey on pivot algorithms in general, we refer to Terlaky and Zhang [11]. Recently, Fukuda, Lüthi and Namiki [5] introduced a class of non-simplex pivot method, known as *admissible pivots*. The finite criss-cross method of Chang, Terlaky and Wang belongs to that class.

Compared to the first finite criss-cross algorithm of Chang, Terlaky and Wang, the new variants presented in this paper seem to enjoy more flexibility in selecting pivot elements at the initial stage. Moreover, the pivoting procedure is less dependent on the artificial orderings such as the indices of the variables. As generally true for the criss-cross pivot method, the new algorithms use admissible pivots as well.

The organization is as follows. In the next section we mention the notation. In Section 3 we introduce the first finite criss-cross type method and two new variants. Section 4 contains a

finiteness proof for the new variants, and Section 5 concludes the paper.

2 Notation

Consider the following standard linear programming problem

$$\begin{array}{lll} (P) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b \\ & & x \geq 0 \end{array}$$

and its dual problem

$$\begin{array}{lll} (D) & \text{maximize} & b^T y \\ & \text{subject to} & A^T y + s = c \\ & & s \geq 0 \end{array}$$

where x and (y, s) are respectively the decision variables for (P) and for (D), and $A \in \Re^{m \times n}$, $b \in \Re^m$ and $c \in \Re^n$.

We assume that the constaint matrix A is full-row rank, i.e. rank $A = m$. Denote

$$A = [a_1, a_2, \cdots, a_n].$$

A basis B is an index set with the property that $B \subseteq \{1, 2, \cdots, n\}$, $\mid B \mid = m$ and

$$A_B := [a_i \mid i \in B]$$

is an invertible matrix. Nonbasis N is the complement of B, i.e. $N := \{1, 2, \cdots, n\} \setminus B$.

Given a basis B we call the following matrix a simplex tableau (or a dictionary, cf. Chvátal [3]) with respect to B:

$-z$	$\cdots$	s_j	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_i	$\cdots$	$\bar{a}_{ij}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

where

$$[\bar{a}_{ij}]_{|B| \times |N|} = A_B^{-1} A_N,$$

$$[s_j \mid j \in N] = c_N^T - c_B^T A_B^{-1} A_N$$

$$[x_i \mid i \in B]^T = A_B^{-1} b$$

and

$$z = c_B^T A_B^{-1} b.$$

To indicate the relation with the basis, in the later part of the paper we will denote x by $x(B)$, and s by $s(B)$. Furthermore, we extend $x(B)$ and $s(B)$ to their full dimensions, i.e. we let

$$x(B)_i := \begin{cases} x(B)_i, & \text{for } i \in B \\ 0, & \text{for } i \in N \end{cases}$$

and

$$s(B)_i := \begin{cases} s(B)_i, & \text{for } i \in N \\ 0, & \text{for } i \in B. \end{cases}$$

It is well known that if $x(B) \geq 0$ then B is called primal feasible; if $s(B) \geq 0$ then B is called dual feasible; if B is both primal and dual feasible then it is optimal.

A pivot method is in general an iterative procedure which updates the basis B at each iteration. The classical simplex method is such type of pivot algorithms that either the primal or the dual feasibility of the basis is preserved in the procedure.

In the next section we shall see some non-simplex pivot method, where the finiteness of the procedure is still guaranteed.

3 The criss-cross method

The key point of the criss-cross method is to select an element $i \in B$ and $j \in N$ without resorting to any type of ratio test. The first finite criss-cross algorithm of Chang, Terlaky and Wang proceeds as follows (see also [2, 9, 12] or [11]).

Smallest subscript criss-cross algorithm (A1)

Step 0 Get a basis B to start with.

Step 1 Let $I = \{i \mid x(B)_i < 0\}$ and $J = \{i \mid s(B)_i < 0\}$.

If $I \cup J = \emptyset$, then B is optimal, stop.

Let $k = \min\{k \mid k \in I \cup J\}$.

If $k \in I$, then go to Step 2; if $k \in J$ then go to Step 3.

Step 2 Let $S = \{j \mid j \in N \text{ and } \bar{a}_{kj} < 0\}$.

If $S = \emptyset$, then the primal problem (P) has no feasible solution, stop.

Otherwise, let $j = \min\{j \mid j \in S\}$. Let $B := B \cup j \setminus k$ and go back to Step 1.

Step 3 Let $T = \{i \mid i \in B \text{ and } \bar{a}_{ik} > 0\}$.

If $T = \emptyset$, then the dual problem (D) has no feasible solution, stop.

Otherwise, let $i = \min\{i \mid i \in T\}$. Let $B := B \cup k \setminus i$ and go back to Step 1.

The following result was shown in [2], [9] and [12].

Theorem 3.1 *Algorithm A1 will terminate in a finite number of iterations.*

Next, we present a new variant of the criss-cross method, incorporating the FILO/LOFI rule of Zhang [13] instead of the smallest subscript rule.

FILO/LOFI criss-cross algorithm (A2)

Step 0 Get a basis B to start with.

Step 1 Let $I = \{i \mid x(B)_i < 0\}$ and $J = \{i \mid s(B)_i < 0\}$.

If $I \cup J = \emptyset$, then B is optimal, stop.

Select $k \in I \cup J$ such that x_k has changed its status (basic/nonbasic) most recently. There are two possibilities to be considered here. First, if none of the variables has ever changed its status, then we have all the freedom to select one pivot element. Second, if there are two variables that have been pivotted on most recently (one in I and the other in J), then we simply break this tie by arbitrarily selecting one to proceed.

After k is selected: If $k \in I$, then go to Step 2; if $k \in J$ then go to Step 3.

Step 2 Let $S = \{j \mid j \in N \text{ and } \bar{a}_{kj} < 0\}$.

If $S = \emptyset$, then the primal problem (P) has no feasible solution, stop.

Otherwise, select $j \in S$ such that x_j has become nonbasic most recently. If none has become basic before, then select one arbitrarily.

Let $B := B \cup j \setminus k$ and go back to Step 1.

Step 3 Let $T = \{i \mid i \in B \text{ and } \bar{a}_{ik} > 0\}$.

If $T = \emptyset$, then the dual problem (D) has no feasible solution, stop.

Otherwise, select $i \in T$ such that x_i has become basic most recently. If none has become nonbasic before, then select one arbitrarily.

Let $B := B \cup k \setminus i$ and go back to Step 1.

The next variant of the criss-cross algorithm uses information of the pivot history as well. In particular, it records the pivot frequency for each variable, and always favors the one that has been selected most often to pivot once more. Its detailed presentation is as follows.

Most-often-selected-variable criss-cross algorithm (A3)

Step 0 Get a basis B to start with.

Step 1 Let $I = \{i \mid x(B)_i < 0\}$ and $J = \{i \mid s(B)_i < 0\}$.

If $I \cup J = \emptyset$, then B is optimal, stop.

Select $k \in I \cup J$ such that x_k has changed its status (basic/nonbasic) most often. As in Algorithm A2, we have two possibilities to consider. First, if none of the variables in $I \cup J$ has ever been a pivoting variable before, then we have all the freedom to select one to pivot. Second, if there are more than one variable that have been selected most often (i.e. with equal pivot frequency), then we simply break this tie by arbitrarily selecting one to proceed.

After k is selected: If $k \in I$, then go to Step 2; if $k \in J$ then go to Step 3.

Step 2 Let $S = \{j \mid j \in N \text{ and } \bar{a}_{kj} < 0\}$.

If $S = \emptyset$, then the primal problem (P) has no feasible solution, stop.

Otherwise, select $j \in S$ such that x_j has been so far selected to pivot most often. If more than one variable have been selected to pivot most often, then select one of them arbitrarily.

Let $B := B \cup j \setminus k$ and go back to Step 1.

Step 3 Let $T = \{i \mid i \in B \text{ and } \bar{a}_{ik} > 0\}$.

If $T = \emptyset$, then the dual problem (D) has no feasible solution, stop.

Otherwise, select $i \in T$ such that x_i has been so far selected to pivot most often. If more than one variable have been selected to pivot most often, then select one of them arbitrarily.

Let $B := B \cup k \setminus i$ and go back to Step 1.

We see that the above algorithms A2 and A3 bear great similarity compared to the first finite criss-cross algorithm A1. But, in some way more flexibility is allowed by these two new algorithms. The main task of this paper is to show that Algorithms A2 and A3 are all finite.

4 Finiteness of the new criss-cross variants

Consider an arbitrary basis B. Let $j \in N$. Denote the primal direction $p(B, j)$ as follows:

$$(p(B,j))_i = \begin{cases} -(A_B^{-1}a_j)_i, & \text{for } i \in B \\ 1, & \text{for } i = j \\ 0, & \text{for } i \in N \setminus j. \end{cases}$$

Clearly, $(p(B,j))_i = -\bar{a}_{ij}$ for $i \in B$.

Denote the dual direction $d(B, i)$ as follows:

$$d(B,i)^T = e_i^T A_B^{-1} A.$$

We see that $d(B,i)_j = \bar{a}_{ij}$ for $j \in N$.

The following lemma plays an important role in proving finiteness of various pivot methods.

Lemma 4.1 *Consider two arbitrary bases B and B', and two indices $i \in B$ and $j \in N$. It holds that*

$$p(B,j) \in Null(A)$$
$$d(B,i) \in Range(A^T)$$
$$x(B) - x(B') \in Null(A)$$
$$s(B) - s(B') \in Range(A^T).$$

The proof for this lemma is elementary, and is omitted here.

Based on Lemma 4.1 we now prove the finiteness for Algorithm A2. We shall remark that the finiteness proof for Algorithm A3 is very much the same. The reader may verify that indeed the proof for Theorem 4.1 presented below works for Theorem 4.2 too.

Theorem 4.1 *The FILO/LOFI criss-cross algorithm A2 is finite.*

Proof.

We shall prove the theorem by contradiction. Suppose that the algorithm does not terminate in finite amount of iterations in general. This means that there exist problem instances for which application of Algorithm A2 can yield an infinite sequence of tableaus. Let (P) be such a problem instance with minimum dimension, i.e. Algorithm A2 will be finite when applied to any problem with fewer rows or columns than that of (P). Now denote $B_1, B_2, \cdots, B_k, \cdots$ be an infinite sequence of bases when Algorithm A2 is applied to (P). The corresponding nonbases are denoted by $N_1, N_2, \cdots, N_k, \cdots$.

It is clear that all the variables will change their basic/nonbasic status in this sequence infinite amount of times due to the assumption that (P) is minimum in size.

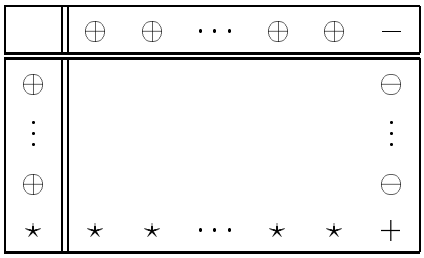
For ease of exposition we call a variable selected in Step 1 of Algorithm A2 an *actively selected variable*, and a variable selected in Steps 2 or 3 of Algorithm A2 a *passively selected variable*.

Now, observe this sequence of tableaus. Consider the first tableau in this sequence such that after this pivot all the variables will have changed their basic/nonbasic status at least once. Let x_t be the last one to be selected among all the variables to pivot. Let the corresponding basis be B.

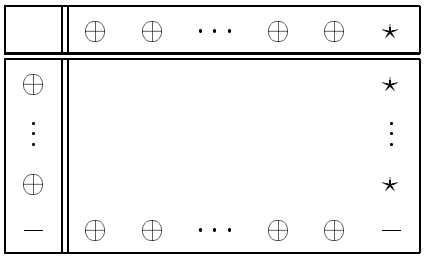
Because the algorithm is completely symmetric in primal and dual, we may assume without loss of generality that x_t is a nonbasic variable to enter the basis B.

There are two possibilities: (1) x_t enters B actively; (2) x_t enters B passively.

These two possibilities correspond to the following tableaus respectively:



and



In these tableaus we denote a positive number by $+$, a non-negative number by $\oplus$, a negative number by $-$, a non-positive number by $\ominus$, and an arbitrary number by $\star$.

We first consider the situation that x_t is selected actively to enter B. Let x_v leaves B passively at the same iteration. There are two separate cases to be considered here. The first case is that $x(B)_v \geq 0$. This case will be treated below. The other case, $x(B)_v < 0$ will be discussed at the end of the proof.

As each variable will enter and leave the basis for an infinite amount of times, there will be a next pivot in which x_t leaves the basis. Consider the first next tableau when x_t is to leave the basis. Let B' be the corresponding basis. There are two possible ways in which x_t leaves the basis: Actively selected to leave, or passively selected to leave. We consider these two cases separately.

Case 1.1: x_t leaves the basis B' actively.

Using Lemma 4.1 we have

$$\begin{aligned} 0 &= \langle s(B) - s(B'), x(B) - x(B') \rangle \\ &= -\langle s(B), x(B') \rangle - \langle s(B'), x(B) \rangle \end{aligned}$$

$$
\begin{aligned}
&= -\sum_{i \in N \cap B'} s(B)_i x(B')_i - \sum_{i \in B \cap N'} s(B')_i x(B)_i \\
&= -s(B)_t x(B')_t - \sum_{i \in N \cap B' \setminus t} s(B)_i x(B')_i - \sum_{i \in B \cap N'} s(B')_i x(B)_i. \qquad (4.1)
\end{aligned}
$$

As x_t is selected actively to leave the basis B', we have

$$x(B')_t < 0. \qquad (4.2)$$

Moreover, as we discussed before, it holds that

$$s(B)_t < 0 \qquad (4.3)$$

and

$$s(B)_i \geq 0 \text{ for } i \neq t. \qquad (4.4)$$

On the other hand, for all $i \in N \cap B' \setminus t$, it must hold that

$$x(B')_i \geq 0 \qquad (4.5)$$

since otherwise we would have chosen x_i to leave the basis instead of x_t, according to the FILO rule.

We know that

$$x(B) \geq 0, \qquad (4.6)$$

and, applying a similar argument as for (4.5) we have

$$s(B')_i \geq 0, \qquad (4.7)$$

for all $i \in B \cap N'$.

Now, using relations (4.2), (4.3), (4.4), (4.5), (4.6) and (4.7), we obtain from (4.1) that

$$0 \leq -s(B)_t x(B)_t < 0. \qquad (4.8)$$

This contradiction proves that **Case 1.1** can never occur.

Now consider the other possibility.

Case 1.2: x_t leaves the basis B' passively.

Let x_k be entering the basis B' actively at that pivot iteration.

Applying Lemma 4.1 in this case as well, we obtain

$$\begin{aligned} 0 &= \langle s(B) - s(B'), p(B', k)\rangle \\ &= \langle s(B), p(B', k)\rangle - \langle s(B'), p(B', k)\rangle \\ &= s(B)_k + s(B)_t p(B', k)_t + \sum_{i \in N \cap B' \setminus t} s(B)_i p(B', k)_i - s(B')_k. \end{aligned} \tag{4.9}$$

We know that $s(B)_i \geq 0$ for all $i \neq t$, $s(B)_t < 0$, $p(B', k)_t < 0$, $s(B')_k < 0$, and

$$p(B', k)_i \geq 0$$

for all $i \in N \cap B' \setminus t$. Therefore, it follows from (4.9) that

$$0 \geq s(B)_t p(B', k)_t - s(B')_k > 0.$$

This contradiction shows that **Case 1.2** cannot happen either.

Now we shall follow the similar line of arguments to treat the case that x_t enters B passively. Let x_l be leaving B actively at the same iteration. (Note that at each pivot iteration there will be always one actively selected variable and one passively selected variable).

Applying the LOFI rule we conclude that

$$d(B, l)_j \geq 0 \text{ for all } j \neq t \text{ and } d(B, l)_t < 0. \tag{4.10}$$

Consider the first next pivot iteration when x_t leaves the basis again. Let the corresponding basis be B'. Similar to the previous analysis we need to consider two separate cases listed below.

Case 2.1: x_t leaves B' actively. And,

Case 2.2: x_t leaves B' passively.

We first consider **Case 2.1**. Applying Lemma 4.1 we have

$$\begin{aligned} 0 &= \langle x(B) - x(B'), d(B, l)\rangle \\ &= x(B)_l - \langle x(B'), d(B, l)\rangle \\ &= x(B)_l - x(B')_l - x(B')_t d(B, l)_t - \sum_{i \in N \cap B' \setminus t} x(B')_i d(B, l)_i. \end{aligned} \tag{4.11}$$

Consider $x(B')$. All variables in $N \cap B' \setminus t$ (and also possibly l) came into B' later than t. Hence, because of the FILO rule we conclude that

$$x(B')_i \geq 0 \tag{4.12}$$

for all $i \in N \cap B' \setminus t$ and $i = l$. Hence we derive from (4.11), (4.10) and (4.12) that

$$0 \leq x(B)_l - x(B')_t d(B,l)_t < 0$$

which is a contradiction. So, **Case 2.1** cannot occur as well. The next case to be considered is **Case 2.2**.

In this situation, let x_r be entering the basis B' actively as x_t leaves B' passively. Now we apply Lemma 4.1 to obtain

$$\begin{aligned} 0 &= \langle p(B',r), d(B,l) \rangle \\ &= p(B',r)_l + d(B,l)_r + p(B',r)_t d(B,l)_t + \sum_{i \in N \cap B' \setminus t} p(B',r)_i d(B,l)_i. \end{aligned} \tag{4.13}$$

Using the FILO rule again we conclude that

$$p(B',r)_i \geq 0$$

for all $i \in N \cap B' \setminus t$ and $i = l$. Certainly, $p(B',r)_t < 0$. Once again we derive a contradiction, based on (4.13) and (4.10), that

$$0 \geq p(B',r)_t d(B,l)_t > 0$$

which shows that **Case 2.2** cannot occur as well.

Finally, we consider the last remaining case: x_t was selected actively to enter the basis B, but $x(B)_v < 0$. This is a peculiar case which will not appear in the analysis of Algorithm A1. Its treatment is as follows.

In this case we know that x_v received the same priority as x_t in that pivot. These two variables swapped their position in some previous pivot. Consider the first next pivot in which x_v is to enter the basis B''. If x_v enters actively, then

$$\begin{aligned} 0 &= \langle s(B) - s(B''), p(B,t) \rangle \\ &= s(B)_t - \sum_{i \in B \cap N'' \setminus v} s(B'')_i p(B,t)_i - p(B,t)_v s(B'')_v - s(B'')_t \end{aligned}$$

which is impossible since

$$s(B)_t < 0,\ p(B,t)_v < 0,\ s(B'')_v < 0$$

and

$$s(B'')_i \geq 0 \text{ and } p(B,t)_i \geq 0$$

for all $i \in B \cap N'' \setminus v$ and $i = t$.

If x_v enters B'' passively and let x_u leaves B'' actively at the same pivot, then

$$\begin{aligned} 0 &= \langle p(B,t), d(B'',u) \rangle \\ &= p(B,t)_u + d(B'',u)_t + p(B,t)_v d(B'',u)_v + \sum_{i \in B \cap N'' \setminus v} p(B,t)_i d(B'',u)_i. \end{aligned}$$

Again this is impossible since

$$p(B,t)_v < 0 \text{ and } d(B'',u)_v < 0$$

and

$$p(B,t)_i \geq 0$$

for all $i \in \{1, 2, \cdots, n\} \setminus v$ and

$$d(B'',u)_i \geq 0$$

for all $i \in B \cap N'' \setminus v$ and $i = t$.

Summarizing, we conclude that the contradiction assumption cannot be true which in turn proves the correctness of the theorem.

Q.E.D.

Analogous to Theorem 4.1 we have the following result.

Theorem 4.2 *Algorithm A3 is finite.*

As we remarked before, Theorem 4.2 can be proven in a nearly identical way as Theorem 4.1, and hence we omit the proof here.

5 Conclusions

A key point behind the finiteness of the three criss-cross algorithms A1, A2 and A3 is that these algorithms work in a recursive manner, in the sense that they always solve a subproblem completely before considering a large subproblem. A general scheme of finite recursive pivot algorithms for linear programming is presented by Jensen [8]. Other remarkable implicit

recursive pivot rules include the rule of Edmonds and Fukuda [4] (for a description, see also [11]).

One advantage of the criss-cross algorithms A1, A2 and A3, compared to general recursive scheme, is that they are simple to implement and easy to remember.

We remark that Fukuda and Matsui [6] gave a different interpretation for the finiteness of the first criss-cross algorithm (A1). It is not clear how to extend a similar interpretation for Algorithms A2 and A3.

Although it is shown that there exists a short path of admissible pivots to optimality (cf. [5]), the challenging open question remains: Is there an implementable polynomial pivot algorithm for linear programming?

Acknowledgement: I would like to thank Tamás Terlaky for his encouragement, and for commenting on early drafts of this paper.

References

[1] Bland, R.G., *New finite pivoting rules for the simplex method*, Math. Oper. Res. 2 (1977) 103-107.

[2] Chang, Y.Y., *Least index resolution of degeneracy in linear complementarity problems*, Tedchnical Report 79-14, Department of Operations Research, Stanford University, U.S.A., 1979.

[3] Chvátal, V., Linear Programming, W.H. Freeman and Company, 1983.

[4] Fukuda, K., *Oriented matroid programming*, Ph.D. Thesis, Waterloo University, Canada, 1982.

[5] Fukuda, K., Lüthi, H.-J. and Namiki, M., *The existence of a short sequence of admissible pivots to an optimal basis in LP and LCP*, Technical Report, Institute for Operations Research, ETH-Zentrum, Zurich, Switzerland, 1996.

[6] Fukuda, K. and Matsui, T., *On the finiteness of the criss-cross method*, European Journal of Operations Research 52 (1991) 119-124.

[7] Fukuda, K. and Terlaky, T., *Criss-cross methods: A fresh view on pivot algorithms*, submitted to Elsevier Preprint, 1997.

[8] Jensen, D., *Coloring and duality: Combinatorial augumentation methods*, Ph.D. Thesis, School of OR and IE, Cornell University, Ithaca, U.S.A., 1985.

[9] Terlaky, T., *A convergent criss-cross method*, Math. Oper. und Stat., ser. Optimization 16 (1985) 683-690.

[10] Terlaky, T., *A finite criss-cross method for oriented matroids*, Journal of Combinatorial Theory (Ser. B) 42(3) (1987) 319-327.

[11] Terlaky, T. and Zhang, S., *Pivot rules for linear programming: A survey on recent theoretical developments*, Ann. of Oper. Res. 46(1993)203-233.

[12] Wang, Z., *A conformal elimination-free algorithm for oriented matroid programming*, Ann. Math. 8(B1), 1987.

[13] Zhang, S., *On anti-cycling pivoting rules for the simplex method*, Operations Research Letters 10 (1991) 189-192.

[14] Zionts, S., *The criss-cross method for solving linear programming problems*, Management Science 15 (1969) 426-445.